11

Ciclos break

Transcripción

[00:00] Ahora, continuando con los ejercicios, vamos a hacer un segundo ejercicio, solamente para dejar un poco más conciso esto del uso de los iteradores. Yo deseo imprimir una matriz. Una matriz es básicamente como una pequeña cuadrícula de X por X. Yo deseo imprimir una matriz de 10 x 10. Entonces voy a crear una nueva clase y vamos a ponerle EjercicioMatriz. Finalizamos.

[00:38] Entonces, como ya sabemos bien, nuestro método Main, perfecto. Para imprimir la matriz de la misma forma que imprimimos esto, que básicamente también es una matriz de 10 x 10, solamente que basado en números múltiplos de cada fila, nosotros vamos a imprimir solamente una matriz de asteriscos. No queremos hacer una operación en este momento, solo queremos asteriscos.

[01:10] Entonces, declaramos un for, el for más grande, que sería entonces int fila igual a 0. Fila que sea menor o igual a 10, y cada vez que él termine de iterar, aumentamos fila. Perfecto. Con fila tenemos una noción un poco más clara de a qué estamos haciendo referencia.

[01:44] Y aquí entramos al siguiente for que sería columna. Filas y columnas. Entonces aquí declaramos nuevamente int columna igual a 0. Columna menor o igual que 10. Y columna ++. Listo. Bajamos esto un poco y aquí adentro solamente vamos a darle un sysout de asterisco, perdón, sería así. De asterisco.

[02:31] Y recordemos como el ejemplo anterior, no debería ser así, debería ser aquí el salto de línea vacío y aquí solamente un system.out.print. Perfecto. ¿I

qué? Nuevamente porque aquí vamos a imprimir todo sucesivamente y aquí vamos a hacer el salto de línea para que salga en este mismo formato. Vamos a agregarle un espacio después de cada asterisco, solamente para que quede un poco más legible.

[03:04] Entonces le damos aquí, pegamos, y le damos un espacio en blanco. Entonces, él imprimiría por cada línea más o menos un espacio como esto. Guardamos, ejecutamos y tenemos aquí nuestra matriz de 10 x 10. Perfecto, una matriz cuadrada perfectamente, bien espaciada. Listo.

[03:31] ¿Qué pasaría si yo deseo ya no una matriz cuadrada sino una matriz digamos triangular? Yo deseo dividir esto en diagonal e imprimir solamente la parte diagonal de aquí abajo, solamente la parte diagonal. Aquí adentro del for yo puedo hacer ifs, puedo ejecutar código aquí tranquilamente.

[04:02] Recordemos cuál es la estructura de if: if, condición, entonces haz algo. Pero ¿qué debería hacer yo para que se detenga este ciclo de imprimir todos los asteriscos? Existe otra palabra reservada en Java llamada break, que es esta de aquí. Con break nosotros rompemos el ciclo, básicamente eso es lo que hace.

[04:36] Break le dice: "Si se cumple esta condición, rompe este ciclo", entonces nunca más va a volver aquí, aún cuando esta condición sea verdadera, él ya no va a ejecutar el ciclo, él aquí está rompiendo el ciclo. Entonces, nuestra condición sería, si es que columna fuera menor o igual que línea, perdón, que fila, si columna es menor o igual que fila entonces rompe el ciclo.

[05:17] Yo aquí ya no quiero imprimir más asteriscos. Si no fuera así, entonces sí, perfecto, sigue imprimiendo asteriscos para ver mi matriz triangular. Vamos a ver qué resultado nos da esto. Imprimimos y no imprimió nada. Vamos a revisar un poco, el error está aquí, el error estuvo aquí en mi condicional, en mi lógica. ¿Por qué?

[05:40] Porque él no debería ser: "si columna es menor o igual a fila", porque esto siempre daría false. Debería ser: "si la columna fuera mayor al número de

fila, entonces ahí rómpelo". ¿Por qué? Porque recordemos que columna estamos refiriéndonos a los ciclos que son impresos horizontalmente, cada uno es una columna, y estamos evaluando al revés, anteriormente.

[06:10] Entonces, guardamos ahí, ejecutamos y vemos claramente que esta matriz triangular fue impresa satisfactoriamente. Ahora, de esta misma forma por ejemplo podemos usar break para otro tipo de condiciones, por ejemplo, si yo necesito que cuando llega determinado punto ya rompa este ciclo, él hace un break.

[06:40] Ahora, como estamos con dos ciclos uno dentro de otro, break no aplica para los dos ciclos al mismo tiempo. Break aplica solamente para un ciclo, para el ciclo donde él está contenido.

[06:55] Entonces si este for fuese detenido por esa sentencia break, break afecta automáticamente aquí, este ciclo es detenido, sale del scope, imprime esta línea y pasa nuevamente al ciclo superior y comienza todo el proceso nuevamente.

[07:19] Entonces, tranquilamente hubiéramos podido ejecutar por ejemplo, si no quisiéramos usar el break para este ejemplo que he dado, recuerden que es la expresión es la expresión booleana, y esto se va a ejecutar si la expresión booleana, si el área de la expresión booleana del for, fuera true. Si nosotros hacemos esto, copiamos aquí, pegamos aquí, no necesitamos esto. ¿Por qué?

[07:54] Porque si esto no fuera true, él no va a imprimir aquí. Cerramos aquí. Vamos a ver si es verdad. Ejecutamos y nuevamente no imprimió nada. ¿Por qué? Porque en este caso nuestra condición booleana aquí está equivocada. Le estamos diciendo que cada vez que columna sea superior a fila, ejecute eso. ¿Pero qué es lo que pasa aquí? ¿Por qué sale el error? ¿Por qué no imprime nada?

[08:25] Porque como comienza con 0, cada vez que llega aquí inicializa la primera vez, el primer recorrido con 0, entonces 0 nunca es mayor que 0 para

el primer caso por ejemplo, o 0 nunca va a ser mayor que 1 o 2 o 3. Entonces esta condición da false y nunca entra a este ciclo de aquí. Lo que necesitamos aquí es invertir la evaluación del signo.

[08:59] Entonces ahora, con la condición correcta, ejecutamos y tenemos el mismo resultado, tenemos nuestra matriz triangular ya impresa en consola. Y es de esta forma como ya hemos aprendido a lo largo de este curso a usar variables, condicionales y ahora ciclos, y estamos aplicándolo aquí. Se va a aplicar en el trabajo el día a día.

[09:27] Entonces yo quisiera motivarlos a ustedes a no dejar de participar en foros, no dejar de practicar ejercicios de programación para que se hagan cada vez más familiares con la forma de la sintaxis de Java, los operadores que maneja Java, yo sé que es un poco más verboso que de repente otros lenguajes de programación, pero cada parte que Java usa tiene un sentido aislado, es completamente cerrado.

[10:01] Es muy fácil cuando tú conoces el lenguaje saber dónde necesitas modificar para corregir un error o añadir un comportamiento que deseas. Como les dije, participen en el foro de aquí de Alura, estamos constantemente respondiendo dudas, si tienen un ejercicio que no pueden resolverlo, compártanlo y podemos aprender todos.

[10:25] Ya en el curso que sigue de aquí, el siguiente curso que es más avanzado, vamos a usar todo esto que hemos aprendido como base para poder realizar digamos operaciones más complicadas como transformar todo un código procedural ya en orientación a objetos, usando cosas un poco más avanzadas del lenguaje Java.

[10:49] Entonces espero tenerlos nuevamente en el siguiente curso y estudien, estudien y estudien. Y practiquen, practiquen mucho. Chau, chau.